Issues in Informing Science and Information Technology Volume 8, 2011

Reinforcing and Enhancing Understanding of Students in Learning Computer Architecture

Cecile Yehezkel
Bar Ilan University,
Ramat Gan, Israel
yehezkc@mail.biu.ac.il

Revital Leibovitch
ORT In the Name of Baron Rothschild, Binyamina, Israel
rleibovi@ort.org.il

Ilya Levin
School of Education, Tel-Aviv University, Tel-Aviv, Israel
Ilia1@post.tau.ac.il

Abstract

The main goal of the research presented in the paper is to study the Electrical Engineering (EE) and Computer Engineering (CE) curriculum in order to improve its coverage, with the goal of better preparing graduates to pursue a wider range of professional careers. The research deals with the differences in studying subjects related to computer architecture by two groups of students having different prior knowledge: EE students and CE students. This research establishes the implications of hands-on studies and considers the gap in knowledge, and whether these studies reduce the gaps. This research was conducted for the same “Microprocessors practicum” course. The first group (CE students) acquired a large body of knowledge in computer architecture, while the second group (EE students) acquired knowledge mainly in “Introduction to programming” prior to the practicum course, and this fact was reflected in a pretest conducted for the two groups. The official test that was taken at the end of the semester, which examined the knowledge that was acquired during the course’s lectures, revealed no difference between the two groups. However, a posttest, specifically arranged for the purpose of this study, strongly indicated that the Computer Engineering students have an advantage over the Electrical Engineering students. Regarding the objective that the lab should provide students with an understanding of computing processes, it seems that lack of prior knowledge hinders it. The lab is meaningful provided that the students are exposed to prior studies, in which case, the lab serves to reinforce the subject matter. At the university in which the research took place, suitable changes in the curriculum have already been made.

Keywords: Computer and Electrical engineering education, Computer Architecture, Microprocessor, Research and Development.

Introduction

Computer Engineering studies combine two different fields of knowledge: hardware and software. Computer Engineering deals mainly with computer architecture and begins with the hardware-software hierarchy level of abstraction from digital logic, whereas Electrical

Engineering deals with hardware below this level of abstraction – from silicon layers, to transistors, logic gate implementation, and digital logic. The border between the two is rarely dealt with. In the integrative concept, one wishes to strengthen the synergism between these two worlds, and to emphasize the importance of studying Computer Architecture to Programming studies, and especially Assembly Language. Curriculum development requires that the developers engage in analysis and decision-making concerning the required number of software and hardware courses studied and concerning the dependency between the two. This is essential in order to define the proper order in acquiring knowledge. As the ACM - Association for Computer Machinery (2004) declared in ACM/IEEE Joint Task Force on Computing Curricula, "Students need to understand computer architecture in order to structure a program so that it runs more efficiently on a real machine. In selecting a system to use, they should be able to understand the trade off among various components, such as CPU clock speed vs. memory size". The aim of the research presented here was to review the curriculum in order to improve its coverage, with the goal of better preparing graduates to pursue a wider range of professional careers, as recommended by McGettrick et al. (2003).

The paper in organized as follows: the rest of this section presents the program of a newly founded Engineering program and focuses on the characteristics of computer architecture topics and teaching approaches. The next section describes the goal of the research and the experimental framework and the third section details the methodology. The results are presented in the fourth section. The final section summarizes the conclusions and examines the perspectives.

Computer Engineering Undergraduate Studies

In this section, the authors present the concept of a new Computer Engineering (CE) program that was implemented in the newly founded School of Engineering at Bar-Ilan University in Israel. The concept includes a description of the basic ideas concerning the structure of the CE Department, its organization, the CE curriculum, issues concerning teaching and research, the infrastructure of the laboratories, ideas of assessment, and organization of the final project.

Establishing a new Engineering Department has always been considered as a challenge. In particular, regarding a Computer Engineering (CE) Department, this challenge is even greater for a number of reasons: (a) the high prestige of computer-related professions in today's society; (b) rapid development of new information technologies and the appearance of corresponding innovative teaching strategies and new learning environments including laboratory infrastructures; (c) the great influence of the modern high-tech industry on the content of the engineering curriculum. At the same time, the challenge is also motivated by the need to distinguish the CE studies from the Computer Science studies, which are widely known and have been intensively developed during the last decades.

More specifically, two main dilemmas have to be resolved when developing a new Computer Engineering Curriculum.

1) The "Theory vs. Practice dilemma" that deals with a compromise between fundamental studies, which are oriented toward basic academic subjects based on classical science and engineering university studies, and industry oriented studies based on rapidly changing computing and information innovations.
2) The "Programming vs. Designing dilemma" that deals with the compromises between programming as the main computing skill and designing as the more subject-oriented professional skill.

Most high-tech companies have R&D programs, manufacture hardware and software, and solve industrial problems by means of technology. Their activities might indeed provide a useful curriculum roadmap. However, to enable students to enjoy productive and meaningful careers, it is necessary to radically change the content of the curriculum of the technology majors (Stephen & Roberts, 2008).

Several curriculum changes and guidelines have been proposed that attempt to address the changes in technology and to design optimal pedagogical approaches in response to these changes (Director et al., 1995; McGettrick et al., 2003, 2008; Meier et al., 2007). In the early 21st century, the Joint Task Force for Computing Curricula identified five main areas of computing degrees: computer engineering, computer science, information systems, information technology, and software engineering (Joint Task Force on Computer Engineering Curricula, 2004).

These areas represent the academic programs that the Joint Task Force believes represent the state of the field and the educational outcomes that students should pursue. They have identified a suite of "computing" and "non-computing" areas that students in each of the five areas should understand. The list of knowledge and skills areas identified by the Joint Task Force that defined the components of the five areas was derived from academic programs and curricula that have evolved over a long period of time between theory and practice, especially since it has an enormous impact on their students' employment prospects. Regardless of how the academic majors and degrees are termed, it is the content of each degree's curriculum that will determine our students' ability to find gainful employment.

The CE program focuses less on alternative programming languages and more on architectures, integration, and interoperability; less on algorithms and discrete structures and more on engineering practices. The CE curriculum was established, having in mind that the number of programming jobs will decline, and will become more specialized and distributed across the globe.

The new CE curriculum is a 4-year program and comprises 178 credit points (200 hours). The curriculum consists of three main parts: Fundamental studies (49 credit points), Computer Science courses (26 credit points), Electrical Engineering courses (38 credit points), Computer Engineering courses (57 credit points), and two elective courses. The curriculum includes 20 hours of labs and a final project. The main features of the presented curriculum are as follows:

1) Fundamental studies, in comparison with the standard Electrical Engineering (EE) program, comprising two additional courses in discrete mathematics.
2) Computer Science courses are distributed during the first three years of the program (one CS course each semester).
3) Electrical Engineering classes are given to the CE students together with the EE students; consequently, the requirements for the CE and EE students are the same.
4) All the CE courses comprise three main clusters:
    a) Fundamental CE courses (Digital Design, Introduction to Computer Engineering, Microprocessors and Assembly language, Computer Architecture, Simulation and Simulation Languages),
    b) Computer Design Courses (Advanced Logic Design, Design of Microcomputers, Hardware Design Methods, Embedded Systems Design), and
    c) Computer Communication courses (Introduction to Coding Theory, Codes for Computer Systems, Introduction to Computer Communication, Topics of Computer Communication, Computer Networking and Communication, and Speedy Networks and Multimedia).

When constructing the curriculum, the authors had two fundamental concepts in mind:

1) Integration of theoretical and practical aspects of CE into the curriculum, while emphasizing the importance of the students having a strong mathematical background. The CE theoretical courses are accompanied by proper labs, which are detailed in table 1.
2) Design orientation. The importance of design as the essence of engineering in general and CE in particular was promoted. The students become familiar with the various facets of designing a computer system. The experimental aspect is performed in the CE teaching labs.

Table 1: CE courses accompanied by proper labs

Course	Theoretical	Lab
Microprocessors and Assembly language	2	2
Micro-Computer Lab	1	3
Micro-Computer Design	3	3
Simulation and Simulation Languages	3	1
Hardware Design of Digital Systems	2	2
Embedded Systems Design	2	2
Capstone graduate project		7

The Complexity of the Computer Architecture Topics

The computer architecture topics are connected to many contiguous domains: digital logic, microprocessors, embedded systems, operating systems, and compiler and programming languages. Nisan and Schocken (2005) illustrated the related topics' hierarchy in the following way: “Of course machine language is also an abstraction -- an agreed upon set of binary codes. In order to make this abstract formalism concrete, it must be realized by some *hardware architecture*. And this architecture, in turn, is implemented by a certain *chip set* -- registers, memory units, ALU, and so on". This interweaving of the related topics reflects the complexity of the computer model that has to be presented at different levels of abstraction. Each level hides the artifacts and mechanisms of the previous lower level. It can be described by a top-down as well as a bottom-up approach (Yehezkel et al., 2007). To encompass the whole domain and deal with its growing complexity, Knuth (2003), in his keynotes, recommended using a bottom-up education approach.

In this hierarchy, the assembly language is the lower programming language and each of its instructions is the translation of one unique machine instruction. One machine instruction is the microcode of hardware activation. Hyde, in his book "the Art of the Assembly Language", wrote that although assembly language has "a pretty bad reputation", it is required to understand the backstage operation of computer processors (Hyde, 2003). It is a means of making the interaction of the hardware (CPU, memory, I/O) and software (the program) comprehensible (Yehezkel. 2003). The importance of learning computer architecture and the difficulties encountered by teachers and students have been well documented (IEEE Micro, 2000; Cassel et al., 2000). Kumar and Cassel (2002) found that many faculty members who teach this subject are teaching outside their areas of specialization and are not entirely comfortable with the task. To improve the learning of computer architecture, instructors have searched for better pedagogical methods and tools (Bem, 2003). Moreover, at the curriculum level (as shown in the previous section), great efforts have to be invested to integrate the theoretical and practical aspects of computer architecture topics. In the present article, the concern was to review the curriculum's courses related to microcomputer architecture topics and to reinforce microcomputer architecture understanding.

Research Goals

Revising a curriculum requires implementing research in order to detect the inadequacies in the coverage of the current program (deficiencies and redundancies). Such a revision has to be based on research findings in order to perform the required curriculum modifications relating to prerequisite courses and course links to deepen and strengthen the coverage in the required core topics.

As shown in the previous section, designing a curriculum is a very difficult task involving hard decision making in an era of an exponentially increasing body of knowledge. This rapid evolution of computer engineering requires, as recommended by Computing Curricula (2001), an ongoing review of the corresponding curriculum.

Furthermore, CE curriculum designers struggle to preserve the department's identity as Computer Engineering (CE); the program is often perceived as a blending of courses from the Electrical Engineering (EE) and Computer Science (CS) fields (Dunne et al., 2007). This issue is even more crucial when designing the curriculum of the computer engineering track during the process of establishing a new school of engineering compounded by the two departments of EE and CE. The courses that can be offered by the two different departments are limited by the sets of faculty, budget restrictions and the low number of freshman applicants. These difficulties only add to the complexity of an inter-department faculty that is responsible for curriculum planning. Moreover, the expertise of the faculty lies on different targets in contiguous domains along the departmental "borders". This situation causes meaningful differences in the requirements of prerequisites, expectations and evaluation metrics between departments (Dunne et al., 2007).

This research was conducted while considering the special circumstances needed to establish an experimental framework that can shed light on the implications of differences in curriculum design. When the research took place, the school of engineering was "young" and the curriculum was established under the constraints of the founding process (core courses, faculty, and infrastructure). In the dynamic process of establishing two tracks in the school of engineering, something unusual happened: third year EE and CE students learned the same course, "Microcomputer" with a different background in the relevant domain. Being involved in the curriculum design, this situation provided us with the possibilities to investigate the implications of the different background on students' course content assimilation, understanding, and programming skills acquisition.

Being aware of the difficulties encountered by students when learning microprocessor topics, the concern was to define the impact of differences between CE and EE students' prior knowledge on their comprehension of microcomputer mechanisms and to investigate if hands-on studies would reduce the knowledge and comprehension gap.

For this purpose the research methodology was to:

a) Determine the prior knowledge of the CE and EE groups of students.

b) Evaluate the level of knowledge that CE and EE groups of students acquire on microcomputer functions during the microcomputer laboratory.

c) Analyze the level of depth in the CE/EE students' understanding of microcomputer mechanisms.

Methodology

The Target Population

The target population consists of two groups of students in their third year at the University. One group studies Computer Engineering (CE), and the second group studies Electrical Engineering (EE). In each group 30-35 students study the same “Microprocessors lab” course consisting of a 1-hour lecture and 3 hours of practice.

The two groups of students studied “Introduction to Computing”, a course that includes programming in “High-Level Language”, writing algorithms, as well as acquaintance with variables and working environments. The CE group also studied three theoretical courses dealing specifically with computer architecture and firmware. The CE group studied the computer model “Bottom-up”, from the basics to the upper level. The EE group had a limited exposure to logic, as shown in the following table:

Table 2: Prerequisite in EE and CE stream for "Microprocessor lab" course

		CE	*EE*
Year 1	*Semester 1*	*Introduction to Computing The C programming Language*	
		Introduction to Computer Engineering	*A course from the EE curriculum**
	Semester 2	*Digital & logic Systems*	
Year 2	*Semester 2*	*Microcomputer & Assembly language*	*A course from the EE curriculum**
		Advanced Logic Design	*A course from the EE curriculum**
		Data structure	*A course from the EE curriculum**
Year 3	*Semester 1*	*Microprocessors lab - ADUC841 processor -*	

* *An EE course disconnected from CE or CS curriculum*

Laboratory Assignments

The “Microprocessors lab” course consists of a one-hour lecture, given to the two groups by the same lecturer (the lesson), followed by three hours of “hands-on” practice (the lab). As preparation for the next lesson and lab, the students have home assignments. The preparation for the lab is individual and requires each student to submit his own report. At the lab they work in pairs. During the lecture, on lab day, the lecturer goes over all subjects necessary for successfully completing the lab assignment. The students are given a second briefing at the beginning of each lab as well. During the “Microprocessors lab” course, the two groups of students learn how to operate an 8051 family Micro-controller, called ADUC841. During lab work, the students wrote programs according to the lecturer’s requirements. The programs are intended to operate the controller and the peripheral components on the ADUC841 card. All students have completed all assignments (the course requirements), as detailed. All pairs of students successfully completed all lab assignments.

Each assignment deals with a different aspect of programming the controller and operating its peripherals. The student needs to run a program that is given in order to understand what it does and how (prior to the lab). In class the students are required to make the necessary changes in the program, which will address the new demands. Each new program derives from the previous one and is the basis for the following program. There is a manual that accompanies the laboratory, which contains all the theoretical background and explanations needed to understand and correctly operate the controller and its peripherals. At the end of each chapter, the students are required to answer some questions that refer to the assignment and to submit a full and detailed report.

The laboratory manual (Engelberg, 2008) contains a comprehensive set of scaffolding assignments that enable the student to learn the microprocessor and its peripherals. The assignments cover the main topics: Programming a delay to blink a LED, using an External Interrupt, timing interrupts with a timer, acquaintance with the programming environment IDE (compilation, debugging, and download), Stack, serial communication with HyperTerminal using UART, Parallel Ports, counters, digital to analog and analog to digital conversion.

The first ten assignments deal with the controller's abilities and its internal parts. The other assignments deal with the peripherals as well, thus allowing for more programming options and uses of the controller. The assignments are detailed and perfectly exemplify how the controller works as well as its limitations. As previously mentioned at the end of each lab, thought-provoking questions are given, with the aim of checking students' understanding and ability to analyze the assignment. Some of these questions are dealt with directly and indirectly in the pretest and the posttest.

Description of the Pretest and Posttest

The pretest was conducted before the course began, with the aim of determining the prior knowledge of each group. The posttest was conducted at the end of course with the intention of determining the level of comprehension. The pretest and the posttest are presented in the Appendix. More details on the research can be found in (Leibovitch, 2007).

Presentation of the Results

Pretest Results

In the pretest, questions 1 to 4 dealt with personal information that shows that the two groups of students (CE and EE) were homogeneous (each student within each group has studied the same courses).

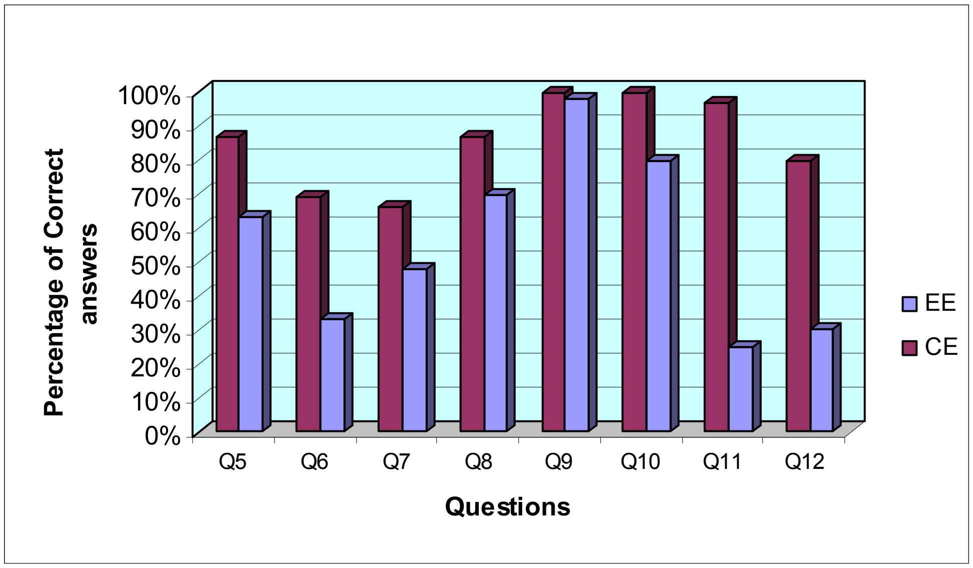


Figure 1. Distribution of correct answers in the pretest

The remaining eight pretest questions are shown in Figure 1 and were divided into 3 groups:

Group I (Questions 5, 11, 12)*:* This group of questions refers to students' Mental Models and the way they grasp the architecture and hierarchy of the computer units. Of the 80% of CE students that drew a correct (schematic) model, 97% (of correct answers) provided a correct literal explanation. Despite the fact that 63% of the EE students drew a correct (schematic) model, only 30% of them provided a correct literal explanation.

Group II (Questions 6, 7, 8)*:* This group of questions determines whether the students know the Memories' functions and distinguishes between the different kinds. The use of memory is an integral part of programming in Assembly, addressing memory, data saving, retrieval, and usage.

In examining the graph in figure 1, we can see that CE students answered these questions more accurately than EE students. This can be related to non-formal (incorrect) knowledge and an individual interpretation.

Group III (Questions 9, 10): This group of questions shows that each group of EE and CE students understand information represented in the computer memory: how data, numbers, and instructions are saved in memory; however, 20% of the EE students do not understand the essence of memory, and cannot differentiate between number and instruction representation in computer memory.

The mean score for CE group (M=85.8, SD=13.3, N= 29) was higher than the mean score for group EE (M=55.9, SD= 26.2, N= 40). The pretest and posttest were voluntary but the final test was compulsory causing slight differences in the number of students in target groups in pretest, final test and posttest.

Posttest Results

The final test that CE and EE students took at the end of the course was essentially targeted to evaluate the level of knowledge that CE/EE students acquire about microcomputer functions during the microcomputer laboratory. There are no meaningful differences between the two groups (M=79.3, SD=12.1, N= 26 for CE group; M=82.3, SD=10.3, N= 42 for EE group). The histogram of grades is presented in Figure 2.

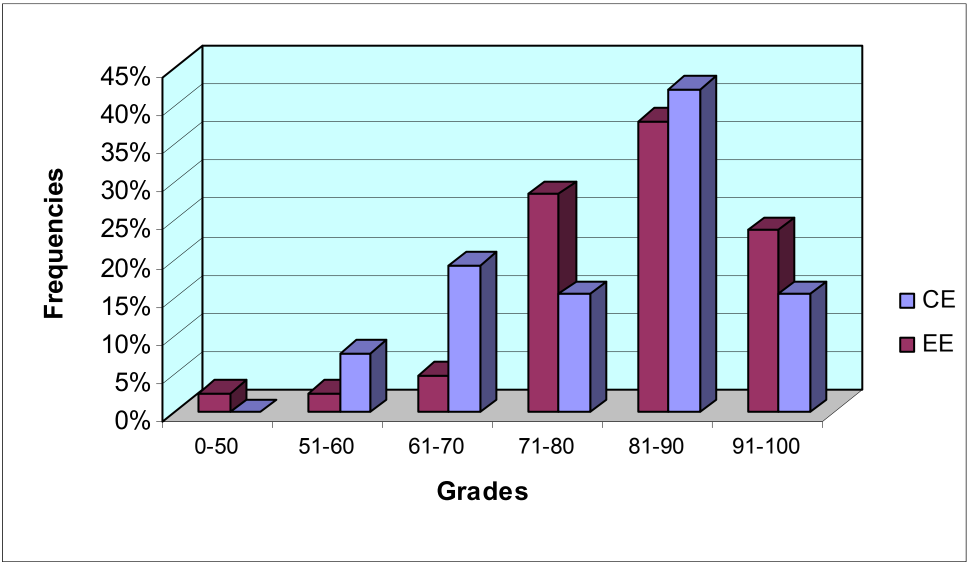


Figure 2. Frequencies of final test scores

The posttest was targeted to evaluate CE/EE students' deep understanding of the microcomputer mechanisms. The distribution of correct answers in the posttest is shown in figure 3. We can see that for each questions in the posttest the percentage of correct answers among CE students is higher than the percentage of correct answers among EE students. The mean score for CE group (M=65.4, SD=22.9, N= 15) is higher than the mean score for group EE (M=48.6, SD= 26.6, N= 35).

Questions 5, 6, and 7 of the posttest refer to specific processes that were dealt with during the course, and relate to formal knowledge learned in class. It can be seen that the “Hands-on laboratory” achieves the same impact in each group independent of prior knowledge except for question 7. In the case of question 7 the knowledge required to question 7 may had created conflicts with the previous knowledge of CE students. Since CE students scored higher, this may imply that having prior knowledge has an advantage.

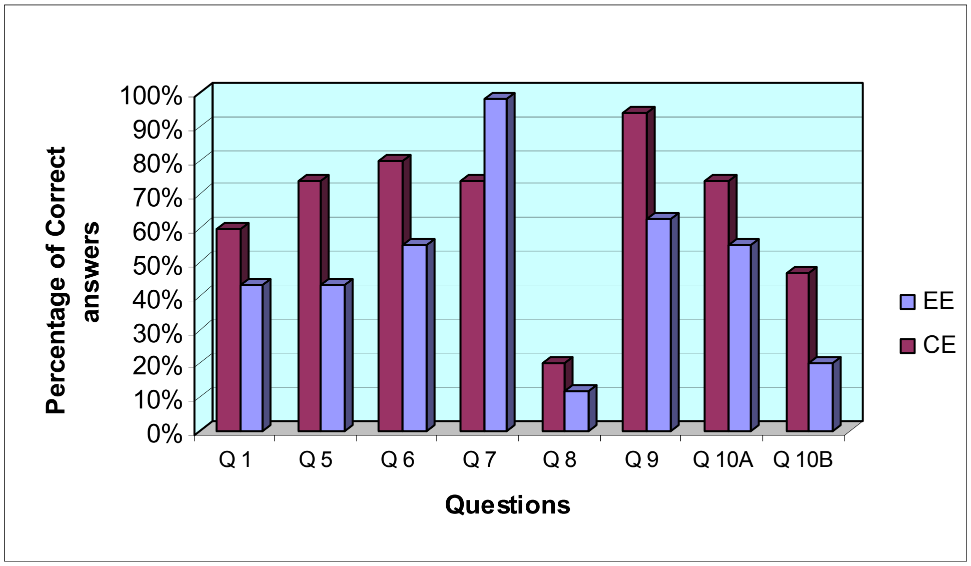


Figure 3. Distribution of correct answers in the posttest

Conclusions and Perspectives

This paper dealt with the research conducted with undergraduate students from two different engineering programs: Electrical Engineering (EE) & Computer Engineering (CE). The aim of the study was to analyze the coverage of computer architecture topics in the Computer Engineering program in comparison with the Electrical Engineering program. The study was conducted using the microcomputer practicum course studied by the two groups: the EE students and the CE students.

The main conclusions of the research can be summarized as follows:

a) Although the students in both the EE and CE streams exhibited similar performances in the final course examination, meaningful differences in the deepness of their understanding of the course material were detected.

b) The narrow theoretical background that was provided to the EE students during a one-hour lecture in the microprocessor course was sufficient to accomplish laboratory tasks and to perform well in the final examination,

c) This narrow theoretical background failed to provide the EE students with an in-depth understanding of the sophisticated microprocessor mechanisms related to computer architecture.

The practicum course was much more fruitful for students owning CE background. In this case it reinforces the subject matter. It provides them with understanding the principles of computer architecture and allows them to apply this body of knowledge to real-world problems and situations (ACM/IEEE Join Task Force on computing curricula). These results led to a revision of the EE curriculum, resulting in an additional course of Computer Architecture as a prerequisite to the Microprocessor practicum course. The necessity of theoretical studies as a background of practical one is proved is our study. The authors plan to further analyze the effects of the change and to pursue formative research on the curriculum toward achieving further improvements.

Acknowledgment

The authors would like to thank Prof. Engelberg who developed the course and the course materials. His support enabled the authors to conduct the research.

References

ACM/IEEE Joint Task Force on Computing Curricula, Overview Report, June 2004.

Bem, E. Z., A case for teaching computer architecture. In Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20 (Adelaide, Australia). T. Greening and R. Lister, Eds. Conferences in Research and Practice in Information Technology Series, vol. 140. Australian Computer Society, Darlinghurst, 2003, Australia, pp. 1-7.

Cassel, L., Kumar, D., Bolding, K., Davies, J., Holliday, M., Impagliazzo, J., "Distributed expertise for teaching computer organization and architecture" (ITiCSE 2000 Working Group Report). ACM SIGCSE Bulletin, Vol. 33, No. 2, 2000, pp. 111 – 126.

Computing Curricula 2001: Report of the ACM/IEEE-Computer Science Joint Curriculum Task Force. Available: http://computer.org/educate/cc2001/

Director, S.W., Khosla. P.K., Rohrer, R.A., and Rutenbar, R.A. "Re-engineering the curriculum: Design and analysis of a new undergraduate electrical and computer engineering Degree at Carnegie Mellon University". Proceedings of the IEEE, 83, 9, 1995, pp. 1246-1269.

Dunne, B.E. Blauch, A.J. Dulimarta, H. Ferguson, R. Sterian, A. Wolffe, G., Work In Progress – CE Curriculum Development Based on IEEE-CS/ACM Body of Knowledge Recommendations. Frontiers in education conference - global engineering: knowledge without borders, opportunities without passports, 2007. FIE '07. 37th annual, 2007, pp. F3H-1-F3H-2.

Engelberg, S., A Microprocessor Laboratory Using the ADuC841, 2008. Available: http://cc.jct.ac.il/~shlomoe/Public/manual.pdf

Hyde, R. The Art of Assembly Language Programming. No Starch Press, San Francisco, CA, Sept. 2003, ch. 6: Memory Architecture. Available: http://webster.cs.ucr.edu/AoA/Windows/HTML/MemoryArchitecture.html.

IEEE Micro, Special Issue on Computer Architecture Education, Vol. 20, No. 30, 2000.

Joint Task Force on Computer Engineering Curricula: IEEE Computer Society/Association for Computing Machinery, "Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering", pp. 30-31. Available: http://www.eng.auburn.edu/ece/CCCE/

Knuth, D. Bottom-up education, keynote in Proceedings of 8th annual conference on Innovation and technology in computer science education ITiCSE'03, Thessaloniki, Greece, ACM Press, 2003, pp. 2-2. Available: http://iticse2003.uom.gr/Iticse_day3

Kumar, D. and Cassel, L., "A state of the course report: Computer organization and architecture". SIGCSE Bulletin, Vol. 34, No.3, 2002, pp. 175 – 177.

Leibovitch, R., "Contribution of Computer Architecture and Microprogramming studies to understanding Computer Processing and Process of Programs Executing ", Master Thesis, School of Education, Tel-Aviv University, March 2007 (in Hebrew).

McGettrick, A., Theys, M. D., Soldan, D.L., and Srimani, P.K., "Computer Engineering Curriculum in the New Millenium", IEEE Transactions on Education, Vol. 46, No. 4, November 2003, pp. 456-462.

Meier, R., Barnicki, S.L., Barnekow, W., and Durant, E., "Work in progress — A balanced, freshman-first computer engineering curriculum", Frontiers in education conference - global engineering: knowledge without borders, opportunities without passports, 2007. FIE '07. 37th annual, October 2007, pp.F3H-17-F3H-18.

Nisan, N. and Schocken, S., The Elements of Computing Systems, Building A Modern Computer from the First principles, MIT Press, 2005, Available: www.idc.ac.il/csd.

Stephen J. A. and Roberts, E., "Point/Counterpoint. Technology curriculum for the early 21st century", Communications of the ACM, Vol. 51, Issue 7, July 2008, pp. 27-32.

Yehezkel, C., Ben-Ari, M. and Dreyfus, T. "The contribution of visualization to learning computer architecture", CSE on Special Issue on Teaching Hardware-software, Vol. 2, No. 17, 2007, pp. 117-127.

Yehezkel, C. (2003). "Making program execution comprehensible—one level above the machine language". SIGCSE Bulletin, Vol. 35, No. 3, pp.124 – 128.

Appendix
Pretest

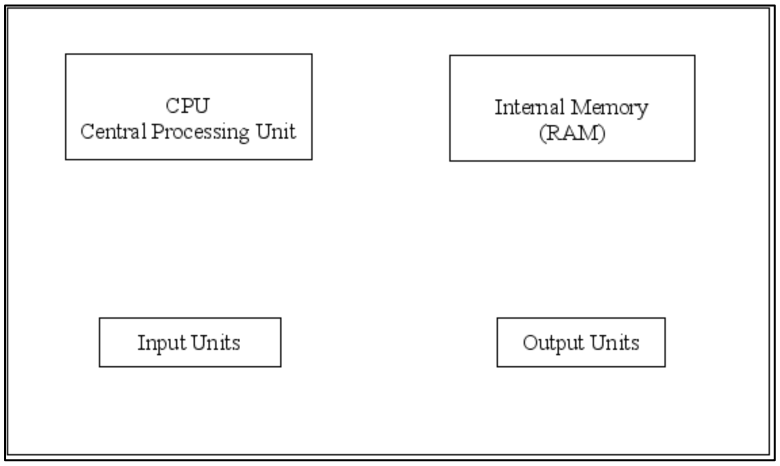
1. Field of Studies: Electronic Engineering / Computer Engineering.

2. Mark √ next to the courses you've taken:
   - ☐ Introduction to Computing
   - ☐ Introduction to Computer Engineering
   - ☐ Logical and Digital Systems
   - ☐ Assembly and Microprocessors
   - ☐ Advance Logic Design
   - ☐ Data Structure
   - ☐ Architecture and Data Structure

3. Have you studied any Microprocessor-related course? YES / NO, where?

4. Do you have any experience in the field of Microprocessors? YES / NO, if the answer is yes, please elaborate:

5. In the following section, the basic units of the computer are presented.
   - o Illustrate, using arrows, the connection between the units.
   - o Describe the function next to each arrow.



Carefully read the following questions and choose the correct answer (there is only one correct answer for each question).

6. What is the computer's internal memory used for (not the CPU cache memory)?
   a) To store data.
   b) To store data and instructions.
   c) As an interface between the processor and the user.
   d) To perform the program's instructions.

7. After the compiling process has been completed and when running, the program is saved in the:
   a) Internal Memory.
   b) Hard Disc.
   c) Cache Memory
   d) Central Processing Unit.

8. The most significant distinction between RAM and ROM is:
   a) RAM memory is larger than ROM memory.
   b) RAM memory is volatile and ROM memory is not.
   c) ROM memory can be written and RAM memory cannot.
   d) There is no significant distinction between the two; both are slower than the Hard Disc.

9. In what form is a number saved in memory?
   a) As Decimal numbers.
   b) As Binary numbers.
   c) As ASCII codes.
   d) As Decimal, Binary, or Hexadecimal numbers, in whatever form the user has typed.

10. In what form are the program instructions saved in memory, when running?
    a) In Assembly language.
    b) As ASCII codes.
    c) In the language it was written.
    d) As Binary codes.

11. Choose the correct sentence:
    a) The length of machine codes is constant.
    b) The length of a machine code depends on the instruction.
    c) The Program Pointer points to the instruction being executed.
    d) The Program Pointer increases by one at the time of execution.
    e) The length of the operation codes is architecture dependent.

12. What are the steps in executing a program?
    a) Executing one instruction after the other as they are written in the program.
    b) Fetching an instruction from memory, decoding the instruction, and executing it, repeated for all program instructions.
    c) Fetching an instruction, followed by the CPU translating the instruction to machine code and executing it, repeated for program all instructions.
    d) Uploading the program from the ROM to the RAM

13. Write the most efficient algorithm (literal) or a block diagram to do multiplication, without the Multiply Instruction.
    Note: $A\cdot B = \underbrace{A+\ldots\ldots+A}_{\text{B times}} = \underbrace{B+\ldots\ldots+B}_{\text{A times}}$

Posttest

Field of Studies: Electronic Engineering / Computer Engineering.

Carefully read the following questions and choose the correct answer (there is only one correct answer to each question).

1. Follow the program code, which adds two numbers. Machine code is provided.

Machine Code	Assembler	Comments
	cseg at 0h	
0000 74 01	mov a, #1h	; Move 1 to A
0002 74	x: db 74h	; Define x as a variable
0003 24 02	add a, #x	; ADD the variable to A

   a) The program cannot be compiled.
   b) The program can be compiled and it does what is needed.
   c) The program can be compiled but it does not do what is needed.

2. If you have chosen (a) in question 1, provide an explanation. (What is the compilation error?)

3. If you have chosen (b) in question 1, what is the added result?

4. If you have chosen (c) in question 1, what does the program actually do?

5. After compiling the program file (for example, sample.asm in μVision3), and downloading the file sample.exe to the ADUC841 card, with aim of RUNNING it, pressing RUN will cause:
   a) The computer processor executes the program instructions saved in the computer memory and activate the ADUC841 card.
   b) The computer processor executes the program instructions saved in the card memory and activate the ADUC841 card.
   c) The card processor executes the program instructions saved in the computer memory and activates the ADUC841 card.
   d) The card processor executes the program instructions saved in the card memory and activates the ADUC841 card.

6. When an Interrupt occurs, the processor stops executing the main program and deals with the appropriate interrupt program. How does the processor "know" how to return to the correct address?
   a) The program counter PC does not change during the interrupt jump, which is why at RETI the processor is in the correct address.
   b) The processor saves the address to be returned to in the stack and provides it to the processor when needed (RETI).
   c) As an Interrupt occurs, the address to return is pushed into the stack, and at the end of an Interrupt it pops out to return to the correct address.
   d) When the Interrupt occurs the return address is saved in the Interrupt Register; at the end of an Interrupt it is copied to the program counter PC.

7. What process is accomplished as RESET is pressed?
   a) The processor immediately stops and jumps to END the instruction.
   b) The processor finishes the present program and begins a new one.
   c) The registers are restarted and the processor jumps to 0000 address.
   d) The registers are restarted and the processor jumps to the beginning of the program.

8. Upon opening a HEX file (executing file), which was made by the µVision3 or ASM51 compiler, in the editor we will find:
   a) Program instructions written in the ADUC841 Assembly language.
   b) Gibberish (nonsense).
   c) Combinations of 1 and 0, which are the machine codes.
   d) Other: ____________________________________.

9. Follow the program code, and show the value of the PC (program counter) in each step (one loop is enough). PC starts at 0000.

PC	CSEG	Machine code		ASM	Comments
1	0000	74 01	START:	MOV A ,#01h	
2	0002	7F 08		MOV R7, #08h	; Move 8 to R7
3	0004	F5 90	NEXT:	MOV P1, A	; Move content of A to P1
4	0006	12 10 00		CALL NANA	; Call for A subroutine
5	0009	23		RL A	; Rotate left
6	000A	DF 04 00		DJNZ R7, NEXT	; Decrease R7 and jump if no zero
7	000D	80 0D 00		JMP $	; Jump here
8	0010	79 FF	NANA:	MOV R1, #0FFH	; Move to R1 FFh
9	0012	7A FF	BABA:	MOV R2, #0FFH	; Move to R2 FFh
10	0014	DA 14 00	SHUV:	DJNZ R2, SHUV	; Decrease R2 and jump if no zero
11	0017	D9 12 00		DJNZ R1, BABA	; Decrease R1 and jump if no zero
12	001A	22		RET	; Return from sub routine

10.
   a) Given two program codes, doing the same function, what do these programs do (in one sentence)? You can assume R0=2 & R1=100.
   b) Which of the two programs is more efficient, why? (Illustrate with calculations).

Program 1

```
STA: CJNE R0, #0, X     ; compare R0 with zero, jump if not equal to x
        JMP SOF                   ; jump
X:      CJNE R1, # , X1  ; compare R1 with zero, jump if not equal to x
        JMP SOF                   ; jump
X1:     CLR A              ; clear A
N:      ADD A, R0                 ; (A+R0) →A
        DEC R1                    ; (R1-1) →R1
        CJNE R1, #0, N   ; compare R1 with zero jump if not equal to x
SOF:    NOP                       ; no operation.
```

Program 2

```
STA:    CJNE R0, #0, X
         JMP SOF
X:      CJNE R1, #0, X1
         JMP SOF
X1:     CLR C   ; 0 → C
         MOV A, R0       ; R0 → A
         SUBB A, R1      ; (A-R1 –CY) →A
         JNB 0E7h, N1    ; jump if MSB in A is 0
         MOV A, R0
         XCH A, R1       ; A ↔ R1
         MOV R0, A
N1:     CLR A
N2:     ADD A, R0
         DEC R1
         CJNE R1, #0, N2
SOF:    NOP
```

Biographies

Cecile Yehezkel holds a M.Sc. in Bio-Medical Engineering from Tel Aviv University and received her Ph.D. degree in Science Teaching from the Weizmann Institute of Science. She is currently the head of instructional laboratories at the School of Engineering at Bar-Ilan University. She leads the Computer Science, Academia & Industry program for talented high school students in the Davidson Institute of Science Education in the Weizmann Institute of Science. She has developed a learning software environment to teach computer architecture. Her research interests focus on simulation design and evaluation, computer architecture and engineering education.

Revital Leibovitch received her M.A degree in Science Teaching from the Tel Aviv University. She also holds a B.Ed.Tech from the Science Teaching Academic College of ORT in Jerusalem. She is currently teaching courses related to microcomputers and low level language and electronics at the ORT Binyamina High School.

Ilya Levin received the M.S. degree in electrical engineering from the Leningrad Transport Engineering Institute, Leningrad, Russia, in 1976, and the Ph.D. degree in computer engineering from the Institute of Electronics, Latvian Academy of Science, Latvia, in 1987.He is currently an Associate Professor with the School of Education, Tel Aviv University, Tel Aviv, Israel, and is the Head of the Department of Mathematics, Science and Technology Education. Prof. Levin is the author more than 120 research papers both in Engineering and in Technology Education. His current research interests include logic design, design automation and technology education.